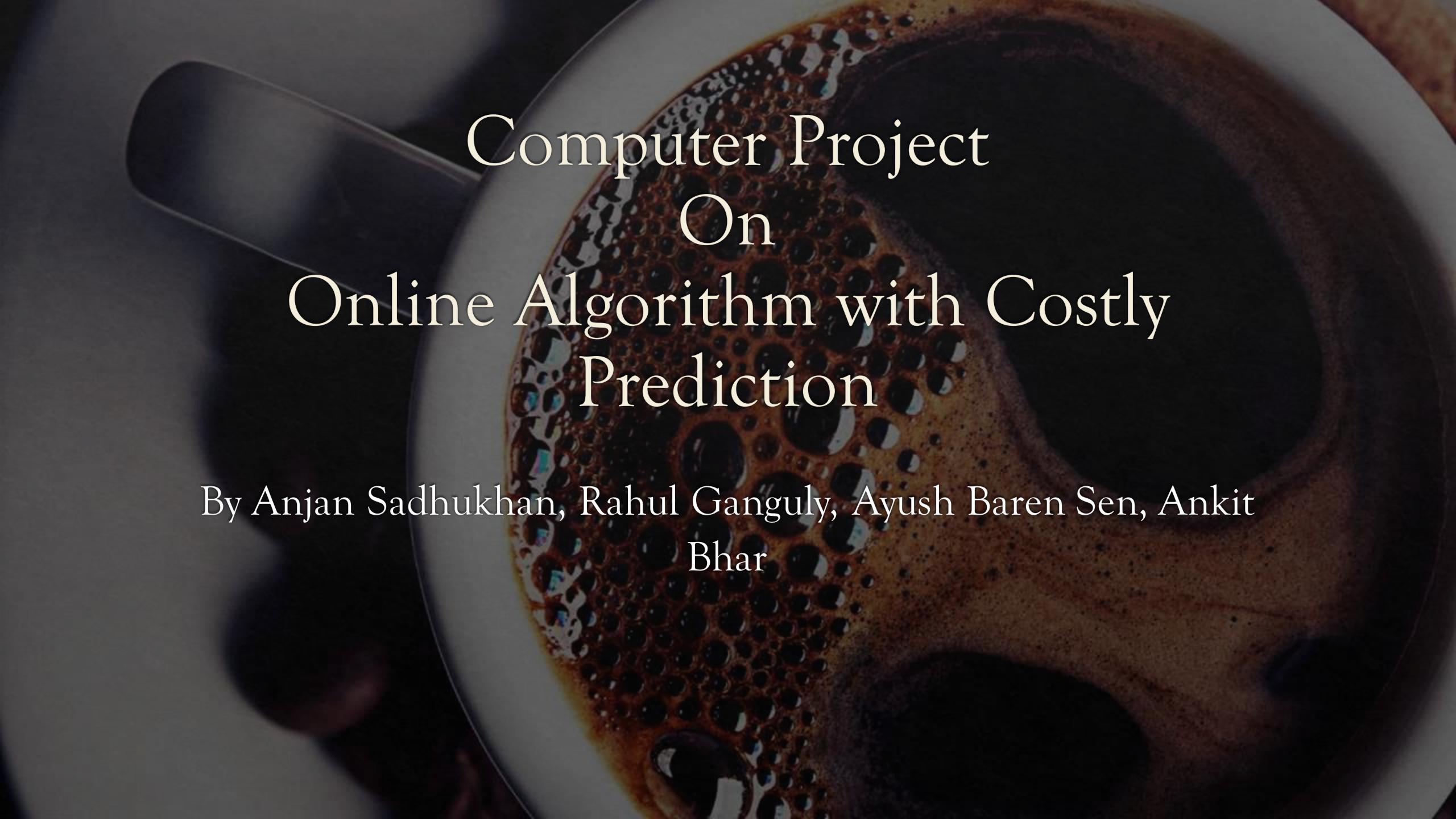

Computer Project
On
Online Algorithm with Costly Prediction

By Anjan Sadhukhan, Rahul Ganguly, Ayush Baren Sen, Ankit Bhar

Question 1

When do we ask the predictor in the ski-rental problem?

Algorithm

- Buy at day b without asking, if $c \geq \theta * b$ (Using deterministic algorithm)
- Ask predictor on day $t* + 1$ and rent if

$$d* < t* + b \text{ and } c < \theta* b$$

- Ask the predictor at day $t* + 1$ and buy immediately if

$$d* \geq t* + b \text{ and } c < \theta* b \text{ and}$$

buy at (1 - c + math.sqrt(((c+1)**2)+4*b*c))/2 day

Here θ is 1 + (math.sqrt(((c+1)**2)+4*c*(b-1))+c-1)/(2*b)

Deterministic Algorithm

- An algorithm for the Ski Rental Problem that purchases the skis on the day $t = b$ is 2-competitive.
- So,

$$\text{ALGO}(I) = \begin{cases} l & \text{if } l < b, \\ (b-1) + b = 2b - l & \text{if } l \geq b \end{cases}$$

where l is the last day of ski season and b is the cost of ski and rent we assume unity.

THE PYTHON CODE IS AS FOLLOWS:

```
import math
def predictor(b,c,r):
    b = int(b)
    c = int(c)
    r = int(r)
    if c == 0:
        print("WE WILL ASK THE PREDICTOR PREDICT THE SKI AT THE FIRST DAY")
    elif r >= b:
        print("WE WILL BUY THE SKI AT THE FIRST DAY")
    elif b < c:
        print("WE WILL BUY THE SKI AT",round(b/r)," th DAY")
    else:
        o = 1 + (math.sqrt(((c+1)**2)+4*c*(b-1))+c-1)/(2*b)
        t = (1 - c + math.sqrt(((c+1)**2)+4*b*c))/2
        if c >= o*b:
            print("WE WILL BUY THE SKI AT",round(b/r)," th DAY")
        else:
            print("WE WILL ASK THE PREDICTOR AT ",round(t),"th DAY")

b = input("ENTER THE COST OF THE SKI:")
c = input("ENTER THE COST OF THE PREDICTOR:")
r = input("ENTER THE COST OF THE RENT:")
predictor(b,c,r)
```

Question 2

Given apriori information about the mean and variance of the true number of ski days, should we ask the predictor?

Value of Prediction

- We consider this simple buy on day 0 or rent forever, which we show is already tight for natural parameters ($\mu = \Theta(b)$ and $\sigma = o(b)$).
- BoR := Buy on day 0, if $\mu \geq b$

  Rent forever, if $\mu < b$

We identify a threshold function $f*(\mu, \sigma, b)$, and show that this is actually the upper bound of BoR. Thus, the threshold function will enable us to decide if we should ask the predictor. That is if c is greater than this threshold value, we use BoR, otherwise we ask the predictor, i.e, use PRED. We call this algorithm VoP (Value of Prediction).

Algorithm

- There exists a threshold function $f*(\mu, \sigma, b)$, such that the algorithm VoP satisfies, $VoP \leq OPT + \min(f*(\mu, \sigma, b), c)$.
- If $\mu = b$ and $\sigma \leq b$, then we have that algorithm VoP satisfies, $VoP \leq OPT + \min(\sigma/2, c)$
- We divide our problem into two subcase:
- Case 1 : When μ >= b ,

  If b >= σ and (b + math.sqrt(b*b - σ*σ)) >= μ :

  The value of thresold function = (math.sqrt(σ*σ + (μ -b)**2)-(μ -b))/2

  Else the value of thresold function = b /(1 + ((μ*μ)/(σ*σ)))
- Case 2 : When μ < b ,

  If b >= σ and (b + math.sqrt(b*b - σ*σ)) >= μ :

  The value of thresold function = (math.sqrt(σ*σ + (b - μ)**2)-(b - μ))/2

  Else the value of thresold function = μ - (b /(1 + ((σ*σ)/(μ*μ))))

THE PYTHON CODE IS AS FOLLOWS:

```
import math
def asking_predictor(u,v,c,b):
    u = int(u)
    v = int(v)
    c = int(c)
    b = int(b)
    if u >= b:
        if  b >= v and  (b + math.sqrt(b*b - v*v)) >= u:
            f = (math.sqrt(v*v + (u-b)**2)-(u-b))/2
        else:
            f = b /(1 + ((u*u)/(v*v)))
    else:
        if   b >= v and u >= (b + math.sqrt(b*b - v*v)):
            f = (math.sqrt(v*v + (b-u)**2)-(b - u))/2
        else:
            f = u - (b/(1 + ((v*v)/(u*u))))


    if c > f:
        print("Don't ask the predictor")
    else:
        print("Ask the predictor")


u = input("ENTER THE MEAN:")
v = input("ENTER THE VARIANCE:")
c = input("ENTER THE COST OF THE PREDICTOR:")
b = input("ENTER THE COST OF A SKI:")
asking_predictor(u,v,c,b)
```

Question 3

In the bahncard problem, how often should we ask the predictor for the true number of trips taken in certain intervals?

Algorithm

- Let B be the cost of the bahncard , T is the validity of the card , N number of travel request come in [0, N*] interval , p* be the cost of per time asking predictor and mL be the cost per the ticket , and we get a discount of $\beta \in [0, 1]$ if we have the bahncard. we get a discount of $\beta \in [0, 1]$.
- The goal is to minimize the total expenditure namely,

$$mB + (p1 + p2 + p3 + ... + pN)$$

where m is the number of cards purchased and pi is the cost at ith day. A solution to the problem can be represented as a possibly empty sequence $\tau = (\tau 1, \ldots, \tau m)$, of time points in $0, \ldots, N - 1$ at which we buy our cards

- We divide the problem into two cases.
- Case 1: If $T >= N^*$, the bahncard problem is equivalent to ski–rental problem and use the deterministic algorithm.
- Case 2: If $T < N^*$,
- Subcase 1 : If $B + \beta c < c + p$, always buy bahncard at the day of request and there is no necessary of predictor.
- Subcase 2 : If $B + \beta c > c$ & $2c > B + 2\beta c$, if $p + Tm < 2c - (B + 2\beta c)$, go to the predictor for round(N^*/T) time otherwise use randomized algorithm to get $(2 - \beta)$ approximation and inductively so on..

Randomized Algorithm

.No deterministic online algorithm for BP(C; T)can be better than (2 − β)-competitive.

THE PYTHON CODE IS AS FOLLOWS:

```
import math
def asking_predictor(b,t,n,m,p,l,c,h):
    b = int(b)
    t = int(t)
    n = int(n)
    m = int(m)
    p = int(p)
    l = int(l)
    c = int(c)
    h = float(h)
    if t >= m:
        if p == 0 and l == 0:
            print("We will ask the predictor at the first day and for whole length")
        elif n*c >= (b + n*h*c) :
            print("We will buy the bahncard at the first day and we have no need to ask the predictor")
        elif b < p:
            print("We will buy the bahncard at",round((b + n*h*c)/n*c),"th day and we have no need to ask the predictor")
        else:
            o = 1 + (math.sqrt(((p + l +1)**2)+4*(p+l)*(b+n*h*c-1))+p+l-1)/(2*(b+n*h*c))
            t = (1 - p + l + math.sqrt(((p+l+1)**2)+4*(b+n*h*c)*(p + l)))/2
            if p + l >= o*(b+n*h*c):
                print("WE WILL BUY THE SKI AT",round((b+n*h*c)/(n*c)),"th DAY")
            else:
                print("WE WILL ASK THE PREDICTOR AT ",round(t),"th DAY and for only one time")

    if t < m:
        if b + h*c < c + p:
            print("We will always buy the bahncard at the day of request and there is no necessary of the predictor")
        else:
            i = 1
            while i < n-1:
                if i*c < b + i*c*h and (i + 1)*c >= b + (i +1)*c*h:
                    if  (i + 1)*c - (b + (i +1)*c*h) >= p + t*l :
                        print("Go to the predictor at the day of request and ask him to predict for the time interval of validity of the card and we have to go to the predictor for "
                              ,round(m/t),"time")
                    else:
                        print("No need to go predictor and use randomized algorithm to determine 2 - h approximation")
                i += 1


b = input("ENTER THE COST OF THE BAHNCARD:")
t = input("ENTER THE VALIDITY OF THR CARD:")
n = input("ENTER THE NUMBER OF TRAVEL REQUEST:")
m = input("ENTER THE LENGTH OF INTERVAL:")
p = input("ENTER THE COST OF PREDICTION PER TIME:")
l = input("ENTER THE COST OF PREDICTION PER INTERVAL:")
c = input("ENTER THE COST OF TICKETS:")
```

THANK YOU